Cpminer 使用手册

download_gb_file.py 2
quality_ctrl.py 4
get_cds.py 8
filter_seq.py 脚本帮助文档 11
select_seq_by_acc.py 脚本帮助文档 14
cds_align.py 脚本帮助文档 18
trim_start_end3.py 帮助文档 21
add_profile.py 帮助文档 23
coverage_and_identity.py 帮助文档 25
rename.py 帮助文档 27
assemble_with.py 帮助文档 30
supermatrix_creator.py 帮助文档 33

`download_gb_file.py` 脚本帮助文档

功能描述

该脚本用于从 NCBI GenBank 数据库批量下载叶绿体基因组文件（`.gb`格式），支持多线程并发和批次请求，优化下载效率并遵守 NCBI API 请求限制（每秒不超过 3 次请求）。

使用前提

1. 安装依赖库：`pip install biopython pandas func-timeout`
2. 在 NCBI 注册邮箱（替换 `your_email@example.com`）

参数说明

参数	缩写	必填	默认值	描述
`--your_email`	`-e`	否	`your_email@example.com`	用于 NCBI API 认证的邮箱
`--input_file`	`-i`	**是**	-	输入文件路径（CSV/XLSX 格式），需包含 `accession` 列
`--output_dir`	`-o`	**是**	-	输出目录路径（自动创建）
`--threads`	`-t`	否	`10`	并发下载线程数
`--batch_size`	`-b`	否	`3`	每批次处理的 accession 数量

输入文件格式要求

- **文件类型**：CSV 或 Excel（`.xlsx`）
- **必需列名**：`accession`（存放 GenBank 序列号，如 `NC_000001.1`）
- 示例内容：列名称 accession 必须包含

CSV

```
accession
NC_000001.1
NC_000002.1
```

运行示例

bash

```
# 基础用法
python download_gb_file.py -i accessions.csv -o ./gb_files -e your_ema
il@example.com

# 使用 20 线程，每批次 5 个 accession
python download_gb_file.py -i data.xlsx -o ./genomes -t 20 -b 5
```

工作流程

1. **读取输入**：解析 CSV/XLSX 文件中的 accession 列表
2. **去重过滤**：跳过输出目录中已存在的`.gb`文件
3. **任务分派**：
   - 将 accession 按批次分割（默认每批 3 个）
   - 多线程并发处理批次任务（默认 10 线程）
4. **速率控制**：自动限制请求频率（≥0.34 秒/次）
5. **下载重试**：失败后自动重试（最多 3 次）
6. **保存结果**：以基础 accession 命名文件（如`NC_000001.gb`）

输出结果

- 目录结构：

  bash

  ```
  output_dir/
  ├── NC_000001.gb
  ├── NC_000002.gb
  └── ...
  ```

- 文件名规范：去除 accession 版本号（如`NC_000001.11` → `NC_000001.gb`）

注意事项

1. **API 限制**：
   - 单次请求最多支持 3 个 accession（受 NCBI 策略限制）
   - 高并发可能导致 IP 被临时封禁，建议线程数≤20
2. **超时机制**：单批次下载超时时间为 10 分钟
3. **错误处理**：
   - 下载失败时自动重试 3 次
   - 最终失败的 accession 会在终端报错

`quality_ctrl.py` 脚本帮助文档

功能描述

该脚本用于对叶绿体基因组文件（`.gb`格式）进行质量控制和重新注释，主要包含两大功能：

1. **质量筛选**：统计基因组特征，筛选不符合质量标准的文件
2. **重新注释**：对问题文件使用 PGA（Plastid Genome Annotator）进行重新注释

使用前提

1. 安装依赖库：`pip install biopython pandas`
2. 安装必要工具：
   - Perl (v5.10+)
   - BLAST+ (v2.6+)
   - PGA.pl (需单独下载)

参数说明

参数	缩写	必填	默认值	描述
`--input_dir`	`-i`	是	-	输入目录（包含.gb 文件）
`--output_dir`	`-o`	是	-	输出目录（保存问题文件和处理结果）
`--cds_threshold`	`-c`	否	`80`	CDS 数量最小阈值
`--ambig_threshold`	`-a`	否	`0.2`	模糊碱基比例最大阈值
`--PGA_path`	`-PGA`	否	-	PGA.pl 脚本路径
`--inverted_repeat`	`-ir`	否	`1000`	最小反向重复区长度
`--pidentity`	`-p`	否	`80`	BLAST 一致性百分比阈值
`--reference`	`-r`	否	-	参考基因组目录（近缘物种）
`--threads`	`-t`	否	`3`	处理线程数

质量筛选标准

1. CDS 数量：低于设定阈值（默认<80）
2. 模糊碱基比例：高于设定阈值（默认>20%）
3. 问题文件处理：
   - 转换为 FASTA 格式保存到输出目录
   - 删除原始 GB 文件

- 生成质量报告 **`gb_info.csv`**

重新注释流程

1. 依赖检查：自动检测 Perl 和 BLAST 路径
2. 文件分组：按线程数创建处理组
3. PGA 注释：使用参考基因组重新注释

perl

```
perl PGA.pl -r [参考目录] -i [输入] -o [输出]
```

4. 结果整合：合并注释结果并清理临时文件

输出文件

1. 质量报告 (`gb_info.csv`)：

csv

```
filename,organism,sequence_length,gene_count,CDS,tRNA,rRNA,unclear_bases,unclear_ratio,error
```

2. 问题文件：转换为 FASTA 格式（`.fasta`）
3. 重新注释结果：新的 GB 文件（`.gb`）

使用示例

bash

```
# 基础质量筛选
python quality_ctrl.py -i ./genomes -o ./qc_output -c 70 -a 0.15 -t 8

# 质量筛选+重新注释
python quality_ctrl.py -i ./genomes -o ./qc_output \
    -PGA ./tools/PGA.pl \
    -r ./reference_genomes \
    -ir 1500 -p 85 -t 10
```

注意事项

1. **重新注释依赖：**
   - 必须提供 **`PGA_path`** 和 **`reference`** 参数

- 参考基因组需为同一科的近缘物种

2. **路径要求：**
   - 输入目录必须包含`.gb`文件
   - PGA.pl 路径需指向可执行脚本
3. **阈值建议：**
   - 完整叶绿体基因组 CDS 数通常>80
   - 高质量基因组模糊碱基比例<5%
4. **资源消耗：**
   - 每个线程需要约 1GB 内存
   - 大样本集建议使用高配服务器

-

`get_cds.py` 脚本帮助文档

功能描述

该脚本用于从叶绿体基因组文件（`.gb`格式）中提取编码序列（CDS）和核糖体 RNA（rRNA）基因，主要功能包括：

1. **基因提取**：从 GB 文件中提取 85 个标准叶绿体基因序列
2. **序列标准化**：统一基因命名并保存为 FASTA 格式
3. **结果统计**：生成基因数量和位置统计报告

核心功能

- 支持提取 85 个叶绿体标准基因（包括 CDS 和 rRNA）
- 自动处理基因命名差异（如"16S ribosomal RNA" → "rrn16"）
- 多线程并行处理提高效率
- 自动清理空结果文件

参数说明

参数	缩写	必填	默认值	描述
`--input_dir`	`-i`	**是**	-	输入目录（包含.gb 文件）
`--output_dir`	`-o`	**是**	-	输出目录（保存基因序列）
`--threads`	`-t`	否	`3`	处理线程数

提取的基因列表

脚本支持提取以下 85 个标准叶绿体基因：

python

```
['accD', 'atpA', 'atpB', 'atpE', 'atpF', 'atpH', 'atpI', 'ccsA', 'cemA',
 'clpP', 'infA', 'matK', 'ndhA', 'ndhB', 'ndhC', 'ndhD', 'ndhE', 'ndhF', 'ndhG', 'ndhH', 'ndhI',
 'ndhJ', 'ndhK', 'petA', 'petB', 'petD', 'petG', 'petL', 'petN', 'psaA', 'psaB', 'psaC', 'psaI',
 'psaJ', 'psbA', 'psbB', 'psbC', 'psbD', 'psbE', 'psbF', 'psbH', 'psbI', 'psbJ', 'psbK', 'psbL',
```

```
 'psbM', 'psbN', 'psbT', 'psbZ', 'rbcL', 'rpl14', 'rpl16', 'rpl2',
'rpl20', 'rpl22', 'rpl23',
 'rpl32', 'rpl33', 'rpl36', 'rpoA', 'rpoB', 'rpoC1', 'rpoC2', 'rps1
1', 'rps12', 'rps14', 'rps15',
 'rps16', 'rps18', 'rps19', 'rps2', 'rps3', 'rps4', 'rps7', 'rps8',
 'rrn16', 'rrn23', 'rrn4.5',
 'rrn5', 'ycf1', 'ycf15', 'ycf2', 'ycf3', 'ycf4', 'ycf68']
```

输出文件

1. **基因序列文件**：
   - 每个基因单独保存为`.fasta`文件（如`rbcL.fasta`）
   - 序列头格式：`>{基因组 ID}:{基因位置}`
2. **统计报告**：
   - `cds_num.csv`：每个基因组中各基因的数量统计

csv

```
accession,organism,accD,atpA,...,cds_num
```

   - `cds_loc.txt`：每个基因的具体位置信息

txt

```
accession   accD    atpA    ...     cds_num
```

使用示例

bash

```
# 基础用法
python get_cds.py -i ./genomes -o ./cds_sequences

# 使用 8 线程加速
python get_cds.py -i ./gb_files -o ./gene_seqs -t 8
```

处理流程

1. **初始化**：
   - 创建输出目录
   - 为每个基因创建空 FASTA 文件
2. **多线程处理**：
   - 每个线程处理一个 GB 文件

- 提取基因序列并追加到对应基因文件
- 记录基因数量和位置信息

3. **结果整合：**
   - 合并所有线程的统计结果
   - 生成 CSV 和 TXT 报告文件
4. **清理：**
   - 删除空结果文件

注意事项

1. **输入要求：**
   - 输入目录必须包含`.gb`文件
   - 建议先运行质量筛选脚本确保文件完整性
2. **线程设置：**
   - 默认 3 线程，可根据 CPU 核心数调整
   - 大样本集建议使用更多线程（8-16）
3. **序列处理：**
   - 自动处理反向互补链
   - 合并多片段基因（如内含子分隔的基因）
4. **基因命名：**
   - 支持常见别名自动转换
   - 未识别基因标记为"unknown"

`filter_seq.py` 脚本帮助文档

功能描述

该脚本用于根据长度过滤编码序列（CDS），通过比较输入序列长度与参考基因长度（被子植物或裸子植物），保留长度在指定范围内的序列。主要功能包括：

1. **长度过滤**：只保留长度在参考长度指定倍数范围内的序列
2. **序列选择**：同一基因在同一基因组中出现多条序列时，仅保留最长序列
3. **结果统计**：生成包含每个基因长度的统计报告

核心功能

- 基于被子植物/裸子植物的参考基因长度进行过滤
- 自动处理同一基因的多拷贝序列（保留最长拷贝）
- 多线程并行处理提高效率
- 生成详细的长度统计报告

参数说明

参数	缩写	必填	默认值	描述
--input_dir	-i	**是**	-	输入目录（包含.fasta 文件）
--output_dir	-o	**是**	-	输出目录（保存过滤后序列）
--ref_orthologous	-r	**是**	-	参考类型：Ang（被子植物），Gym（裸子植物）
--lower_bound	-lb	否	0.5	长度下限（参考长度的倍数）
--upper_bound	-ub	否	2.0	长度上限（参考长度的倍数）
--threads	-t	否	3	处理线程数

过滤规则

1. **长度范围**：参考长度 × 下限 ≤ 序列长度 ≤ 参考长度 × 上限

2. **重复序列处理**：
    - 同一基因组中同一基因出现多条序列时
    - 仅保留最长的一条序列参与长度筛选

参考基因长度

脚本内置两类植物的标准基因长度：

1. **被子植物 (Angiosperm)**：

```
Python

{'accD':1599, 'atpA':1524, ... 'ycf68':234}
```

2. **裸子植物 (Gymnosperm)**：

```
python

{'accD':1041, 'atpA':1524, ... 'ycf68':240}
```

输出文件

1. **过滤序列**：
    - 每个基因保存为单独 FASTA 文件（如 rbcL.fasta）
    - 序列头格式保持不变
2. **统计报告** (length.csv)：

```
csv
accession,organism,accD,atpA,...
```

    - 记录每个基因组中各基因的最终序列长度
    - 未通过过滤的基因留空

使用示例

```
bash
# 被子植物参考，默认范围(0.5-2.0)
python filter_seq.py -i ./cds -o ./filtered_cds -r Ang

# 裸子植物参考，自定义范围(0.6-1.8)
```

```
python filter_seq.py -i ./gene_seqs -o ./filtered_genes -r Gym
-lb 0.6 -ub 1.8 -t 8
```

注意事项

1. **输入要求：**
    - 输入目录必须包含 cds_num.csv 文件（由 get_cds.py 生成）
    - FASTA 文件名必须与参考字典中的基因名一致（如 rbcL.fasta）
2. **参数范围：**
    - 下限建议≥0.5，上限≤2.0（可调整）
    - 线程数根据 CPU 核心数设置
3. **特殊处理：**
    - 未知基因（不在参考字典中）会被跳过
    - 空结果文件自动忽略

`select_seq_by_acc.py` 脚本帮助文档

功能描述

该脚本用于选择每个物种的代表叶绿体基因组，主要功能包括：

1. **物种名标准化**：根据提供的物种名对照表校正学名
2. **代表基因组选择**：对同一物种的多个基因组，选择 CDS 总长度最长的作为代表
3. **序列重命名**：在序列描述中添加标准化物种名

核心功能

- 自动检测并处理同一物种的多个基因组
- 支持外部物种名对照表校正学名
- 多线程并行处理基因文件
- 生成物种-基因组对应关系表

参数说明

参数	缩写	必填	默认值	描述
--input_dir	-i	**是**	-	输入目录（包含过滤后的 CDS 文件）
--output_dir	-o	**是**	-	输出目录（保存代表基因组序列）
--file_organism_name	-f	否	-	物种名对照表（CSV 格式）
--threads	-t	否	3	处理线程数

处理流程

1. **数据准备**：
    - 读取 length.csv 文件（由前序脚本生成）
    - 计算每个基因组的 CDS 总长度
2. **物种名标准化**：
    - 使用外部对照表校正学名（如提供）
    - 将空格替换为下划线（如 Arabidopsis_thaliana)
3. **代表基因组选择**：
    - 按物种分组
    - 选择 CDS 总长度最长的基因组作为代表

4. **序列提取：**
    - 多线程处理每个基因文件
    - 仅保留代表基因组的序列
    - 在序列头添加标准化物种名

特殊处理机制

1. **重复基因组处理：**
    - 当检测到同一物种有多个基因组时
    - 交互式询问用户是否保留最长 CDS 基因组
    - 用户可选择终止程序（输入 n）或继续（输入 y）
2. **物种名标准化：**
    - 优先使用对照表中的名称
    - 缺失时保留原始名称
    - 自动替换空格为下划线

输入输出示例

输入文件结构：

```
bash
input_dir/
├── atpA.fasta
├── rbcL.fasta
├── ...
└── length.csv  # 必需文件
```

输出文件结构：

```
bash
output_dir/
├── atpA.fasta       # 仅含代表基因组序列
├── rbcL.fasta
├── ...
└── organism_nuique.csv  # 物种-基因组对应表
```

物种名对照表示例 (file_organism_name.csv)：

```
csv
```

```
accession,organism
NC_000932.1,Arabidopsis thaliana
NC_037304.1,Oryza sativa
```

序列头格式变化：

```
diff
- >NC_000932.1:123..456
+ >NC_000932.1:123..456|Arabidopsis_thaliana
```

使用示例

```
bash
# 基础用法（无物种名校正）
python select_seq_by_acc.py -i ./filtered_cds -o ./representati
ve_seqs

# 使用物种名对照表+8 线程
python select_seq_by_acc.py -i ./genes -o ./species_reps \
    -f species_names.csv -t 8
```

输出文件说明

1. **基因序列文件：**
    - 保留原始文件名（如 rbcL.fasta）
    - 仅包含每个物种的代表序列
    - 序列头添加标准化物种名
2. **物种表** (organism_nuique.csv)：

    ```
    csv
    accession,organism,cds_num,length
    NC_000932.1,Arabidopsis_thaliana,85,12345
    ```

    - accession：基因组编号
    - organism：标准化物种名
    - cds_num：基因数量
    - length：CDS 总长度

注意事项

1. **必需输入文件：**
    - 输入目录必须包含 length.csv（由 filter_seq.py 生成）
    - 基因文件必须是 FASTA 格式（.fa, .fas, .fasta）
2. **物种名对照表：**
    - 必须包含 accession 和 organism 两列
    - 未提供的物种将使用原始名称
3. **线程设置：**
    - 默认 3 线程，大样本集建议增加
    - 每个线程处理一个基因文件

`cds_align.py` 脚本帮助文档

功能描述

该脚本使用 MAFFT 对编码序列（CDS）进行多序列比对，主要功能包括：

1. **多序列比对**：对每个基因的同源序列进行精确对齐
2. **并行处理**：支持多进程同时处理多个基因
3. **命令验证**：自动验证用户输入的 MAFFT 命令格式
4. **路径检测**：智能检测系统中 MAFFT 的安装位置

使用前提

1. **必需安装 MAFFT**：
   - Linux/macOS：sudo apt install mafft 或 brew install mafft
   - Windows：下载预编译版本并添加至 PATH
2. **输入文件要求**：
   - FASTA 格式序列文件（.fasta, .fas, .fa）
   - 每个文件包含同一基因的不同物种序列

参数说明

参数	缩写	必填	默认值	描述
--input_dir	-i	**是**	-	输入目录（包含 CDS 文件）
--output_dir	-o	**是**	-	输出目录（保存比对结果）
--para_file	-p	否	3	并行处理的文件数

MAFFT 命令格式要求

用户需提供 MAFFT 运行参数模板，必须包含以下占位符：

1. in.fasta：输入文件占位符
2. out.fasta：输出文件占位符
3. 标准格式：mafft [选项] in.fasta > out.fasta

有效命令示例：

```
bash
mafft --auto --thread 4 --reorder in.fasta > out.fasta
mafft --localpair --maxiterate 1000 in.fasta > out.fasta
```

工作流程

1. **路径检测**：
   - 自动查找系统 PATH 中的 MAFFT
   - 未找到时提示用户输入路径
2. **命令验证**：

    - 检查命令格式是否正确
    - 验证选项是否被 MAFFT 支持
3. **任务分配：**
    - 按基因文件创建比对任务
4. **并行比对：**
    - 使用多进程同时处理多个基因
    - 实时显示进度和错误信息
5. **结果保存：**
    - 比对结果保存为同名 FASTA 文件
    - 保持原始序列头信息

使用示例

```
# 基础用法
python cds_align.py -i ./cds_sequences -o ./aligned_cds

# 高级用法（8 进程+自定义参数）
python cds_align.py -i ./genes -o ./aligned_genes -p 8
```

交互步骤示例：

1. 自动检测 MAFFT 路径（或手动输入）
2. 输入 MAFFT 命令模板：
   ```
   bash
   mafft --auto --thread 4 --reorder in.fasta > out.fasta
   ```
3. 脚本验证后执行比对

输出结果

输出目录结构：

```
bash
output_dir/
├── atpA.fasta       # 比对后的 ATP 合酶基因
├── rbcL.fasta       # 比对后的 Rubisco 大亚基基因
└── ...              # 其他基因比对结果
```

序列头格式保持不变：

```
fasta
>NC_066001:[24468:25851](-)|Achillea_ageratum
ATGGT...TGACA
```

性能建议

1. **线程设置：**
    - 小数据集（<50 基因）：默认 3 进程

- 大数据集（>100 基因）：8-16 进程

2. **MAFFT 参数选择：**
   - 快速比对：--auto
   - 高精度：--localpair --maxiterate 1000
3. **内存管理：**
   - 每个进程需要 500MB-1GB 内存
   - 大基因（如 ycf1）需要更多内存

注意事项

1. **文件命名：**
   - 输入文件必须按基因命名（如 rbcL.fasta）
   - 不支持包含空格的文件名
2. **命令限制：**
   - 必须包含 in.fasta 和 out.fasta 占位符
   - 必须使用>重定向输出
3. **错误处理：**
   - 无效命令会提示重新输入
   - 失败的比对会显示错误信息

`trim_start_end3.py` 帮助文档

功能概述

该脚本用于多序列比对文件的双重修剪：

1. **边界修剪**：去除序列两端 gap 比例超过阈值的未对齐区域
2. **内部修剪**（可选）：使用 trimAL 工具去除序列内部的未对齐片段

使用前提

1. **Python 环境**：
   - Python 3.x
   - 必需库：biopython, argparse
   - 安装命令：pip install biopython argparse
2. **外部依赖**（仅当使用内部修剪时）：
   - trimAL 工具（官方 GitHub）
   - 需通过-tp 参数指定可执行文件路径（如/usr/bin/trimal）

命令行参数

参数	缩写	必选	默认值	说明
--input_dir	-i	是	-	输入目录（存放 FASTA 格式比对文件）
--output_dir	-o	是	-	输出目录（保存修剪结果）
--boundaries_threshold	-bt	否	0.025	边界 gap 比例阈值（0-1 之间）
--trimal_path	-tp	否	-	trimAL 可执行文件路径（启用内部修剪）
--para_file	-p	否	3	并行处理文件数（CPU 核心数上限）

使用示例

```
bash
# 基础用法（仅边界修剪）
python trim_start_end3.py -i input_fasta_dir -o trimmed_results -bt 0.03

# 完整处理（边界+内部修剪）
python trim_start_end3.py -i input_dir -o output_dir -tp /opt/trimal/bin/tr
imal -p 4

# 使用 8 个并行进程
python trim_start_end3.py -i alignments -o results -p 8
```

处理流程

1. **边界修剪**（所有文件必须执行）：

    - 从序列起始位置扫描，去除 gap 比例 > 阈值的列
    - 从序列末端反向扫描，去除 gap 比例 > 阈值的列
    - 输出保留的核心区域序列
2. **内部修剪**（当指定-tp 时触发）：
    - 使用 trimAL 的-automated1 模式处理
    - 自动检测并移除内部未对齐区域
    - 覆盖边界修剪的输出文件

输出说明

- 输出文件名与输入文件名保持一致（如 gene1.fasta → gene1.fasta）
- 所有输出文件保存在--output_dir 指定目录

性能提示

- 并行处理：通过-p 参数调整并行文件数（建议≤CPU 物理核心数）
- 典型耗时：100 叶绿体基因组的提取文件耗时 10 秒（3 核 CPU）

`add_profile.py` 帮助文档

功能概述

该脚本用于将新的序列文件合并到已有的多序列比对结果中，使用 MAFFT 工具实现 profile 比对合并，保留原始序列头信息。

使用前提

1. **Python 环境：**
   - Python 3.x
   - 必需库：biopython, argparse
   - 安装命令：pip install biopython argparse
2. **外部依赖：**
   - MAFFT 比对工具（官方文档）
   - 支持自动检测系统 PATH 中的 mafft 或 mafft.bat，未找到时会提示用户输入路径

命令行参数

参数	缩写	必选	默认值	说明
--input_dir	-i	是	-	存放已有比对结果的目录（FASTA 格式）
--add_profile	-a	是	-	待合并的新序列目录（FASTA 格式）
--para_file	-p	否	3	并行处理文件数（默认值：%(default)s）
--thread	-t	否	1	MAFFT 每个任务使用的线程数(默认值:%(default)s)

注意：输入目录和待合并目录中的文件必须**同名对应**（如 gene1.fasta 需在两个目录都存在）

使用示例

```
bash
# 基础用法（自动检测 MAFFT 路径）
python 8add_profile.py -i aligned_results -a new_sequences

# 指定并行参数（4 个并行任务，每个 MAFFT 任务用 2 线程）
python 8add_profile.py -i base_aligns -a additional_seqs -p 4 -t 2
```

处理流程

1. **路径检测：**
   - 自动搜索系统 PATH 中的 MAFFT 可执行文件
   - 若未找到，提示用户手动输入有效路径
   - 验证输入/输出目录存在且非空
2. **文件匹配：**

    - 扫描两个目录中同名的 FASTA 文件（扩展名：.fasta, .fas, .fa）
    - 创建任务队列（每个任务包含一对匹配文件）
3. **并行合并：**
    - 使用 MAFFT 命令：mafft --reorder --add <新序列> <已有比对> > 输出
    - 临时文件处理确保原子性操作
    - 成功合并后覆盖原始比对文件

输出说明

- 合并结果直接覆盖--input_dir 中的原始文件
- 保留原始序列头信息，仅扩展序列内容

技术细节

1. **MAFFT 参数：**
    - --reorder：保持输出序列顺序与输入一致
    - --add：profile 比对模式（将新序列添加到现有比对）
2. **错误处理：**
    - 文件不存在/空文件检测
    - MAFFT 路径有效性验证
    - 临时文件异常处理
3. **性能优化：**
    - 多级并行（文件级并行 + MAFFT 任务线程）
    - 临时文件同目录存储减少 I/O 开销

典型场景

- 进化分析中新增物种序列合并到已有比对
- 宏基因组研究中分批处理样本数据
- 大规模测序项目的增量式比对更新

提示：当处理大量文件时，建议设置-p 参数为 CPU 物理核心数，-t 参数为 2-4（根据内存大小调整）

`coverage_and_identity.py` 帮助文档

功能概述

该脚本用于分析多序列比对结果，计算每条序列相对于参考序列（第一条序列）的覆盖度和一致性，并根据阈值过滤低质量序列，同时生成详细的统计报告。

使用前提

1. **Python 环境：**
   - Python 3.x
   - 必需库：biopython, pandas, numpy
   - 安装命令：pip install biopython pandas numpy
2. **输入文件要求：**
   - FASTA 格式的多序列比对文件（扩展名：.fasta, .fas, .fa）
   - 每个文件的第一条序列作为参考序列
   - 序列必须已进行多序列比对（含 gap 符号"-"）

命令行参数

参数	缩写	必选	默认值	说明
--input_dir	-i	是	-	输入目录（存放比对后的 FASTA 文件）
--output_dir	-o	是	-	输出目录（保存过滤结果和统计报告）
--threshold	-t	否	0.75	覆盖度/一致性阈值（0-1 之间）
--para_file	-p	否	3	并行处理文件数（默认值：%(default)s）

注意：阈值的设置直接影响结果严格性，建议根据数据类型调整（如：高度保守基因可用 0.9，多样性较高基因可用 0.6）

使用示例

```
bash
# 基础用法（使用默认阈值 0.75）
python coverage_and_identity.py -i aligned_cds -o filtered_results

# 自定义阈值（覆盖度/一致性≥80%）
python coverage_and_identity.py -i alignments -o output -t 0.8

# 使用 8 个并行进程
Python coverage_and_identity.py -i input_dir -o output_dir -p 8
```

指标概念解释

1. **覆盖度（Coverage）：**
   覆盖度 = (序列在参考序列非 gap 位点上的有效碱基数) / (参考序列总长度)
   - 衡量序列在参考序列区域上的完整性

- 排除 gap 位置的影响

2. **一致性（Identity）：**

   一致性 =（序列与参考序列匹配的碱基数）/（序列在参考序列区域上的有效碱基数）

   - 衡量序列与参考序列的相似度
   - 仅在双方都有碱基的位置比较

输出结果

1. **过滤后的 FASTA 文件：**
   - 保留所有通过阈值检查的序列
   - 文件名与输入文件相同
   - 示例：gene1.fasta → gene1.fasta
2. **统计报告（CSV 格式）：**
   - 每个输入文件生成单独报告
   - 文件名格式：<原文件名>_stats.csv
   - 包含四列数据：

列名	说明
description	序列描述信息
coverage	覆盖度（4 位小数）
identity	一致性（4 位小数）
filter_res	过滤结果（PASS/FAIL）

处理流程

1. **读取比对文件：**
   - 以第一条序列作为参考序列
   - 计算参考序列有效长度（排除 gap）
2. **计算指标：**
   - 遍历每条序列的每个位置
   - 仅考虑参考序列的非 gap 位置
   - 统计匹配/不匹配碱基数量
3. **过滤序列：**
   - 双重条件：覆盖度 ≥ 阈值 AND 一致性 ≥ 阈值
   - 未通过序列被丢弃
4. **输出结果：**
   - 生成过滤后的 FASTA 文件
   - 生成包含所有序列指标的 CSV 报告

`rename.py` 帮助文档

功能概述

该脚本用于预处理 FASTA 文件，提供两种核心功能：

1. **导出序列信息**：根据自定义格式解析 FASTA 记录头，提取关键信息并生成汇总表
2. **修正生物注释**：使用校正表批量修正 FASTA 文件中的生物体名称

使用前提

1. **Python 环境**：
   - Python 3.x
   - 必需库：biopython, pandas
   - 安装命令：pip install biopython pandas
2. **输入文件要求**：
   - FASTA 文件必须为**对齐后**的序列（所有序列长度相同）
   - 记录头格式需统一且符合用户定义的格式规范
   - 修正模式需要额外的 CSV 校正表（包含 accession 和 organism 列）

命令行参数

参数	缩写	必选	默认值	说明
--input	-i	是	-	输入路径（文件或目录）
--output_dir	-o	是	-	输出目录
--para_file	-p	否	3	并行处理文件数（仅目录模式有效）

操作模式

运行脚本后需选择操作类型：

```
bash
Select operation:
1 - Export records information   # 导出序列信息
2 - Fix organism annotations     # 修正生物注释
0 - stop execution               # 终止执行
```

使用示例

```
bash
# 导出目录中所有 FASTA 的序列信息
python 10rename.py -i ./fasta_dir -o ./output -p 4
# 操作选择 1，输入格式：例如 "accession:location|organism|description"

# 修正单个 FASTA 文件的生物体名称
python 10rename.py -i input.fasta -o ./corrected
# 操作选择 2，输入格式同上，指定 CSV 校正表路径
```

格式字符串说明

格式字符串定义 FASTA 记录头的解析规则：

- **字段名**：预定义字段包括：
  - accession（必选）
  - location（必选）
  - organism（必选）
  - description（可选）
  - other information（可选）
- **分隔符**：任意字符组合
- **示例**：

```
python
"accession:location|organism|description| other information "
# 解析记录头："NC_12345: join{[83632:84023](-), [82533:82967] |Arabidopsi
s_thaliana|Chloroplast|Sample other information "
# 结果：
#   accession: "NC_12345"
#   location: " join{[83632:84023](-), [82533:82967] "
#   organism: "Arabidopsis thaliana"
```

处理流程

1. **导出序列信息模式：**
   - 解析所有 FASTA 文件的记录头
   - 提取 accession 和 organism 字段
   - 生成去重的 CSV 汇总表（seq_info.csv）
   - 严格检查重复项（accession 和 organism 均需唯一）
2. **修正生物注释模式：**
   - 加载 CSV 校正表（需包含 accession 和 organism 列）
   - 根据 accession 匹配原始记录
   - 生成新记录头格式：>{accession}:{location}|{organism}
   - 输出修正后的 FASTA 文件（保留原始文件名）

输出文件

1. **导出模式：**
   - seq_info.csv：包含两列数据

列名	说明
accession	序列编号
organism	生物体名称

2. **修正模式：**
   - 与输入同名的 FASTA 文件，记录头格式统一为：

```
fasta
>{accession}:{location}|{organism}
```

注意事项

1. **关键检查：**
    - 目录模式：自动验证 FASTA 文件是否对齐（序列长度一致）
    - 导出模式：确保 accession 和 organism 字段无重复值
    - 修正模式：CSV 校正表必须覆盖所有 accession
2. **错误处理：**
    - 发现重复 accession 或 organism 时终止运行
    - 记录头解析失败时显示详细错误信息
    - 格式字符串缺少必选字段时报错

`assemble_with.py` 帮助文档

功能概述

该脚本用于合并两个不同来源的基因序列数据集（CDS 序列和小片段基因序列），根据指定的模式选择最优序列，构建整合的基因矩阵。

使用前提

1. **Python 环境：**
   - Python 3.x
   - 必需库：biopython, pandas, numpy
   - 安装命令：pip install biopython pandas numpy
2. **输入文件要求：**
   - FASTA 文件必须使用统一格式的序列头：>{accession}:{organism}|...
   - 每个 FASTA 文件代表一个基因的所有物种序列
   - 同个基因文件中，每个物种只能有一条序列

命令行参数

参数	缩写	必选	说明
--input_CDS_dir	-c	是	CDS 序列目录（.fasta 文件）
--input_other_gene_dir	-g	是	其他基因序列目录（.fasta 文件）
--output_dir	-o	是	整合结果输出目录
--mode	-m	是	序列选择模式（1/2/3）

序列合并模式说明

模式	选择策略	适用场景
1	优先保留 CDS 序列	CDS 数据质量更高时
2	优先保留小片段基因序列	小片段基因数据更完整时
3	保留最长的序列	需要最大化序列信息时

使用示例

```
bash
# 模式 1：优先保留 CDS 序列
python assemble_with.py -c cds_sequences -g other_genes -o merged_output -m
 1

# 模式 3：保留最长序列
python assemble_with.py -c chloroplast_genes -g nuclear_genes -o combined -
m 3
```

处理流程

1. **输入验证：**
    - 检查 CDS 目录和其他基因目录是否存在
    - 创建输出目录（如不存在）
2. **文件分类处理：**
    - CDS 独有文件
    - 其他基因独有文件
    - 共同文件
3. **序列合并策略：**
    - **独有文件**：直接复制到输出目录
    - **共有文件**：
        - 模式 1：CDS 序列覆盖其他基因序列
        - 模式 2：小片段基因覆盖 CDS 序列
        - 模式 3：选择同物种中最长的序列
4. **序列选择算法：**

```
python
if mode == 1:
    combined = pd.concat([df1, df2])  # CDS 优先
    result = combined.drop_duplicates('organism', keep='first')
elif mode == 3:
    combined = pd.concat([df1, df2])
    result = combined.loc[combined.groupby('organism')['length'].idxmax
()]  # 长度优先
```

输出结果

- 输出目录包含整合后的所有基因 FASTA 文件
- 文件命名与输入文件保持一致
- 每个基因文件中：
    - 物种名称唯一
    - 序列头格式：>{accession}|{organism}|...
    - 序列内容为最优选择结果

注意事项

1. **关键要求：**
    - 同个基因文件中，每个物种必须唯一（脚本会自动去重）
    - 序列头必须包含 accession 和 organism 字段
    - 所有序列必须预先对齐（长度一致）
2. **典型应用场景：**
    - 叶绿体基因组与核基因组的整合
    - 不同测序平台的序列合并
    - 公共数据库与实验数据的结合
3. **性能优化：**
    - 使用 Pandas 进行高效数据操作

    - 按基因文件并行处理（非显示参数）
    - 内存友好型流式处理

提示：模式选择直接影响结果质量，建议根据数据类型选择：

- 保守基因 → 模式 1（CDS 优先）
- 高度变异基因 → 模式 3（长度优先）
- 填补缺失数据 → 模式 2（其他基因优先）

`supermatrix_creator.py` 帮助文档

功能概述

该脚本用于将多个基因的比对序列合并为系统发育分析所需的超矩阵（supermatrix），同时生成分区文件，适用于构建大规模系统发育树。

使用前提

1. **Python 环境：**
   - Python 3.x
   - 必需库：biopython
   - 安装命令：pip install biopython
2. **输入文件要求：**
   - FASTA 格式的多序列比对文件（已对齐）
   - 序列头格式：>{任意 ID}|其他信息…|{物种名}（物种名必须位于最后一个|之后）
   - 同个基因文件中，每个物种只能有一条序列

命令行参数

参数	缩写	必选	默认值	说明
--work_dir	-w	是	-	工作目录（包含所有输入 FASTA）
--output_phy	-o	否	"supermatrix.phy"	输出 PHYLIP 文件名
--partition_file	-pf	否	"partitionFile.txt"	输出分区文件名
--patterns	-p	否	["*.fasta"]	文件匹配模式（支持通配符）

使用示例

```
bash
# 基础用法
python supermatrix_creator.py -w ./alignments -o supermatrix.phy -pf partit
ions.txt

# 自定义文件模式
python supermatrix_creator.py -w gene_data -p "*.fas" "*.fasta" -o combine
d.phy
```

输出文件说明

1. **PHYLIP 格式超矩阵：**
   - 第一行：物种数 总位点数
   - 后续行：物种名（10 字符对齐） + 拼接序列

- 示例：

```
text
3 15
Species1  ATGCTAGCTAGCTAG
Species2  ATG---GCTAGCTAG
Species3  ATGCTAGCTAG----
```

2. **分区文件**：
   - 记录每个基因在超矩阵中的位置范围
   - 格式：{基因名} = {起始位置}-{结束位置}
   - 示例：

```
text
gene1 = 1-300
gene2 = 301-650
```

处理流程

1. **准备阶段**：
   - 切换到工作目录
   - 扫描匹配指定模式的所有 FASTA 文件
   - 验证文件有效性
2. **数据收集**：
   - 提取所有物种名称（基于序列头格式）
   - 记录每个基因的长度
   - 计算总位点数
3. **序列拼接**：
   - 为每个物种创建完整长度的空序列
   - 按基因顺序填充有效序列
   - 缺失数据用 gap（-）填充
4. **输出生成**：
   - 写入 PHYLIP 格式超矩阵
   - 生成分区文件记录基因位置

关键特性

1. **智能物种处理**：
   - 自动识别序列头中的物种名（最后一个|后内容）
   - 处理不同基因中物种的缺失情况
2. **全面统计**：
   - 显示处理的基因文件列表
   - 报告每个基因的长度
   - 计算总位点数和物种数
3. **错误处理**：
   - 跳过无效文件并显示警告
   - 空文件检测
   - 路径验证

注意事项

1. **序列头格式要求：**
    - 必须包含|分隔符
    - 物种名必须位于最后一部分
    - 示例：>GeneID123|其他信息…|Arabidopsis_thaliana
2. **文件要求：**
    - 所有输入文件必须已对齐（序列长度一致）
    - 同基因文件中物种名必须唯一
    - 文件扩展名需匹配指定模式
3. **输出格式：**
    - PHYLIP 格式兼容主流系统发育软件（RAxML, IQ-TREE 等）